МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТРАНСПОРТА»

Кафедра «Учетные системы и технологии бизнес-менеджмента»

КУРСОВАЯ РАБОТА

по дисциплине «Современные технологии проектирования информационных систем»

на тему «Программное средство «Электронная обучающая система» (тренинг по SQL)»

Выполнил
студент группы ГЭ-43 (ГИ-41)
Хроменков Ю. А.

Проверил
к.т.н., доцент
Быченко О. В.

Гомель 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире по мере развития информационных технологий возрастает и роль баз данных.

Остро встают вопросы хранения больших объемов данных без ущерба их корректности таким образом, чтобы при этом минимизировать затраты; предоставления доступа к ним с наименьшими временными задержками, имитируя работу в режиме реального времени; проектирования структуры для хранения информации, отражающей связи, существующие между данными в действительной предметной области, максимально корректно и подробно, но, при этом, учитывая потребности, определяемые условием решаемой задачи; инкапсуляции и сокрытия сложных для понимания процессов обработки и хранения данных и предоставления необходимых для пользователя функций с помощью простого и понятного для человека интерфейса и т. п.

Именно для удовлетворения вышеперечисленных потребностей в современном информационном обществе существует и стремительно и непрерывно развивается такое направление науки о данных (Data Science), как теория баз данных.

В этой области научного познания уже много лет устойчиво занимают свою значимую нишу реляционные базы данных, которые, несмотря на неэффективность и даже невозможность применения при решении некоторых задач, являются самым распространенным на сегодняшний день видом баз данных, ведь они характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

С реляционными базами данных непосредственно и связан SQL – язык структурированных запросов (Structured Query Language). Популярностью описанного вида баз данных обусловлена известность и широта использования SQL, что указывает на актуальность изучения этого языка.

В автобусе по пути на работу, в коридоре на перерыве перед экзаменом или в очереди в поликлинику можно уделить время изучению SQL, а возможность сделать это удобно и эффективно способно предоставить соответствующее мобильное приложение, ведь в современном мире сложно даже представить человека, выходящего из дома без телефона. Но важно, чтобы это приложение не ограничивалось поверхностной основой, а содержало структурированную и понятную информационную составляющую.

Таким образом, объектом данного исследования являются мобильные приложения, а предметом – языки программирования, библиотеки и алгоритмы, позволяющие реализовать мобильное приложение для изучения SQL. Цель – описать данное приложение (его функционал и возможности, пользу, структуру, ключевые отличия от аналогов).

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Краткая характеристика баз данных

База данных (БД) – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Чтоб собирать и анализировать информацию, надо уметь ее сохранять. Конечно, можно сохранять информацию в печатном виде в обычных папках или в Excel-файлах и т. п. Многие компании до сих пор так сохраняют свою информацию, но такой способ подойдет только для маленьких компаний с небольшим количеством данных. Когда компания вырастает, то и необходимой для сохранения информации становится больше, такие варианты сохранения информации становятся непригодны в связи с негибкостью масштабирования и сложностью проведения анализа хранящихся данных. Тогда на помощь приходят базы данных.

Ключевой элемент при работе с базами данных – система управления базами данных (СУБД). Именно СУБД предоставляет пользователю большую часть полезного для пользователя функционала.

Система управления базами данных, сокр. СУБД (англ. Database Management System, сокр. DBMS) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных. Это комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД.

Основные функции СУБД:

- управление данными во внешней памяти (на дисках);
- управление данными в оперативной памяти с использованием дискового кэша;
- журнализация изменений, резервное копирование и восстановление базы данных после сбоев;
- поддержка языков БД (язык определения данных, язык манипулирования данными).

Обычно современная СУБД содержит следующие компоненты:

- ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию;
- процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода;
- подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД;

• сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

• По стратегии работы с внешней памятью системы управления базами данных подразделяются на СУБД с непосредственной записью и с отложенной записью.

В СУБД с непосредственной записью все изменённые блоки данных незамедлительно записываются во внешнюю память при поступлении сигнала подтверждения любой транзакции. Такая стратегия используется только при высокой эффективности внешней памяти.

В СУБД с отложенной записью изменения аккумулируются в буферах внешней памяти до наступления любого из следующих событий:

• Контрольная точка.

• Нехватка пространства во внешней памяти, отведенного под журнал: СУБД создаёт контрольную точку и начинает писать журнал сначала, затирая предыдущую информацию.

• Останов: СУБД ждёт, когда всё содержимое всех буферов внешней памяти будет перенесено во внешнюю память, после чего делает отметки, что останов базы данных выполнен корректно.

• Нехватка оперативной памяти для буферов внешней памяти.

Такая стратегия позволяет избежать частого обмена с внешней памятью и значительно увеличить эффективность работы СУБД.

Часто понятие «базы данных» понимается двояко: говоря о базах данных имеется в виду совокупность БД и СУБД. Это связано с тем, что именно СУБД предоставляет весь функционал, с которым пользователь работает непосредственно и который воспринимается как наиболее важный.

Базы данных в связи с СУБД помогают справиться с большим количеством проблем, решить которые папкам и Excel-файлам не под силу:

• Базы данных позволяют обрабатывать, хранить и структурировать намного большие объёмы информации, чем таблицы – миллиарды и триллионы записей;

• Объём информации в базах данных может быть огромным и не влиять на скорость работы (в Google Таблицах уже после нескольких сотен строк или тысяч символов страница будет загружаться очень медленно);

• Базы помогают защищать данные – они позволяют давать доступ к данным только определенному кругу лиц. При этом можно ставить ограничения, кому к каким данным можно давать доступ и какого типа доступ, только чтение или редактирование тоже;

• Базы данных могут помогать следить за правильностью данных с помощью различного вида проверок;

• Также, базы данных могут позволять большому количеству людей одновременно взаимодействовать с данными. Удалённый доступ и система запросов позволяет множеству людей одновременно использовать базы данных. С электронными таблицами тоже можно работать онлайн всей

командой, но системы управления базами данных делают этот процесс организованнее, быстрее и безопаснее.

Удобство использования баз данных основано на их свойствах:

1. Быстродействие

Современные БД проектируются по принципу «получить данные прямо сейчас», чтобы пользователь не ждал отклик на запрос.

2. Простота получения и обновления данных

Какой бы высокой ни была скорость, это бессмысленно, если нужно сделать много сложных операций, чтобы получить, обновить или добавить данные в базу.

3. Независимость структуры

Изменения в любом количестве и качестве информации не должны влиять на структуру базы данных. Также изменения не должны касаться программного обеспечения и средств хранения, например, жёсткого диска.

4. Стандартизация

Аналогично свойству независимости структуры: при обновлении программного обеспечения или СУБД база данных не должна менять свою структуру или свойства.

5. Безопасность данных

Для каждой категории пользователей делают список ограничений и доступов, согласно которым можно взаимодействовать с информацией из БД.

6. Интегрированность

Данные должны быть логически связаны. И эти связи должны прослеживаться по структуре таблиц.

7. Многопользовательский доступ

Удалённо вносить изменения и получать информацию из БД могут сразу несколько человек с разных устройств.

История возникновения и развития технологий баз данных может рассматриваться как в широком, так и в узком аспекте.

В широком смысле понятие истории баз данных обобщается до истории любых средств, с помощью которых человечество хранило и обрабатывало данные. В таком контексте упоминаются, например, средства учёта царской казны и налогов в древнем Шумере (4000 г. до н. э.), узелковая письменность инков – кипу, клинописи, содержащие документы Ассирийского царства и т. п. Следует помнить, что недостатком этого подхода является размывание понятия «база данных» и фактическое его слияние с понятиями «архив» и даже «письменность».

История баз данных в узком смысле рассматривает базы данных в традиционном (современном) понимании. Эта история начинается с 1955 года, когда появилось программируемое оборудование обработки записей. Программное обеспечение этого времени поддерживало модель обработки записей на основе файлов. Для хранения данных использовались перфокарты.

Оперативные сетевые базы данных появились в середине 1960-х. Операции над оперативными базами данных обрабатывались в

интерактивном режиме с помощью терминалов. Простые индексно-последовательные организации записей быстро развились к более мощной модели записей, ориентированной на наборы.

Сам термин база данных (англ. database) появился в начале 1960-х годов, и был введён в употребление на симпозиумах, организованных компанией SDC в 1964 и 1965 годах, хотя понимался сначала в довольно узком смысле, в контексте систем искусственного интеллекта. В широкое употребление в современном понимании термин вошёл лишь в 1970-е годы.

Чаще всего базы данных классифицируют в зависимости от того, как в них структурирована информация и как с ней взаимодействовать, то есть по используемой при организации БД модели данных. Это, пожалуй, одна из наиболее важных классификаций баз данных, согласно которой можно выделить следующие виды БД:

1. Иерархические

Иерархические БД используют иерархическую модель данных – это модель данных, где используется представление базы данных в виде древовидной (иерархической) структуры, состоящей из объектов (данных) различных уровней.

Рисунок 1 демонстрирует пример структурирования информации о видах транспорта с помощью иерархической модели данных.

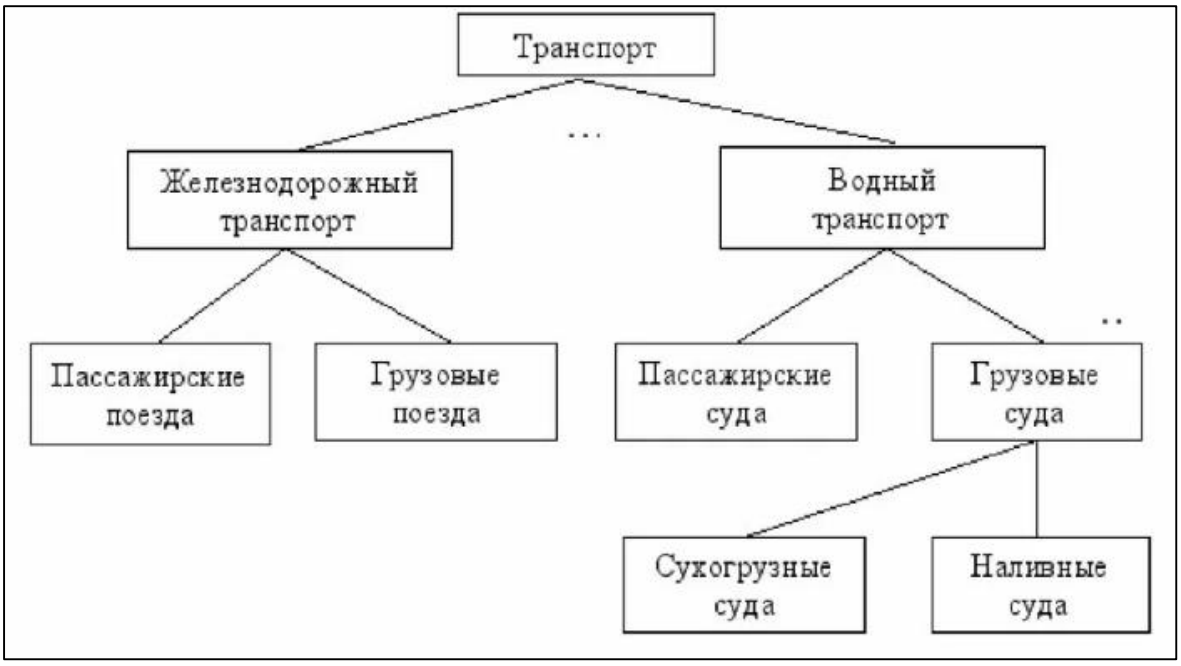

Рисунок 1 – Пример представления информации с помощью иерархической модели данных

Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок имеет несколько потомков, тогда как у объекта-потомка обязателен только один предок. Объекты, имеющие общего предка, называются близнецами (в

программировании применительно к структуре данных дерево устоялось название братья).

Базы данных с иерархической моделью одни из самых старых, и стали первыми системами управления базами данных для мейнфреймов.

Иерархической базой данных является, например, файловая система, состоящая из корневого каталога, в котором имеется иерархия подкаталогов и файлов.

2. Сетевые

Сетевая модель данных – логическая модель данных, являющаяся расширением иерархической.

Разница между иерархической моделью данных и сетевой состоит в том, что в иерархических структурах запись-потомок должна иметь в точности одного предка, а в сетевой структуре данных у потомка может быть любое число предков.

Достоинством сетевой модели данных является возможность эффективной реализации по показателям затрат памяти и оперативности.

Недостатком сетевой модели данных являются высокая сложность и жесткость схемы БД, построенной на её основе. Поскольку логика процедуры выборки данных зависит от физической организации этих данных, то эта модель не является полностью независимой от приложения. Другими словами, если необходимо изменить структуру данных, то нужно изменить и приложение.

На рисунке 2 представлен простой пример структурирования данных с помощью сетевой модели для отражения связей между преподавателями в школе и классами, у которых они ведут (в одном классе ведут несколько преподавателей, каждый – свой предмет).

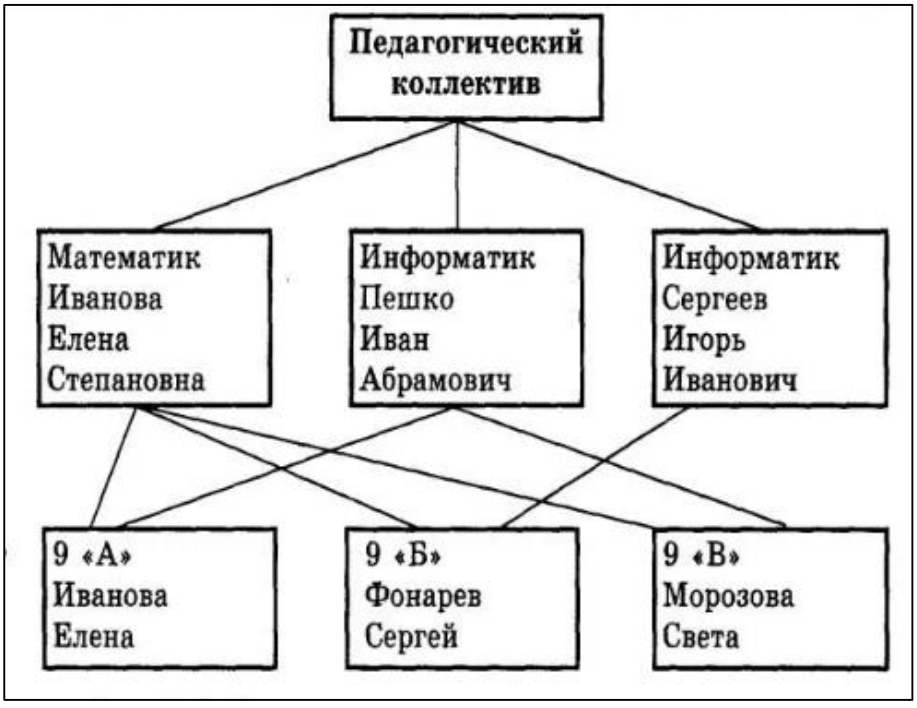

Рисунок 2 – Пример представления информации с помощью сетевой модели данных

3. Объектно-ориентированные

Объектно-ориентированная база данных (ООБД) – база данных, в которой данные моделируются в виде объектов, их атрибутов, методов и классов.

Первые публикации об объектно-ориентированных базах данных появились в середине 80-х годов.

Объектно-ориентированные базы данных обычно рекомендованы для тех случаев, когда требуется высокопроизводительная обработка данных, имеющих сложную структуру.

На рисунке 3 представлен пример организации структуры данных с помощью объектно-ориентированной модели (на рисунке представлена т. н. диаграмма классов в нотации UML).

Рисунок 3 – Пример представления информации с помощью объектно-ориентированной модели данных

В манифесте ООБД предлагаются обязательные характеристики, которым должна отвечать любая ООБД. Их выбор основан на 2 критериях: система должна быть объектно-ориентированной и представлять собой базу данных.

Обязательные характеристики:

- Поддержка сложных объектов;
- Поддержка индивидуальности объектов;
- Поддержка инкапсуляции;
- Поддержка типов и классов;
- Поддержка наследования типов и классов от их предков;
- Перегрузка в сочетании с полным связыванием;
- Вычислительная полнота;
- Набор типов данных должен быть расширяемым.

Необязательные характеристики:

- Множественное наследование
- Проверка типов
- Распределение
- Проектные транзакции

4. Реляционные

В таких базах данных используется реляционная модель данных (РДК) – логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики, как теория множеств и логика первого порядка.

Использование реляционных баз данных было предложено доктором Коддом из компании IBM в 1970 году.

Реляционная модель данных включает следующие компоненты:

- Структурная составляющая – данные в базе данных представляют собой набор отношений.
- Аспект целостности – отношения отвечают определённым условиям целостности. РМД поддерживает декларативные ограничения целостности уровня домена (типа данных), уровня отношения и уровня базы данных.
- Аспект обработки (манипулирования) – РМД поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление).

На рисунке 4 представлен пример схемы реляционной базы данных для хранения информации о прайс-листах и связанных с ними объектами предметной области.

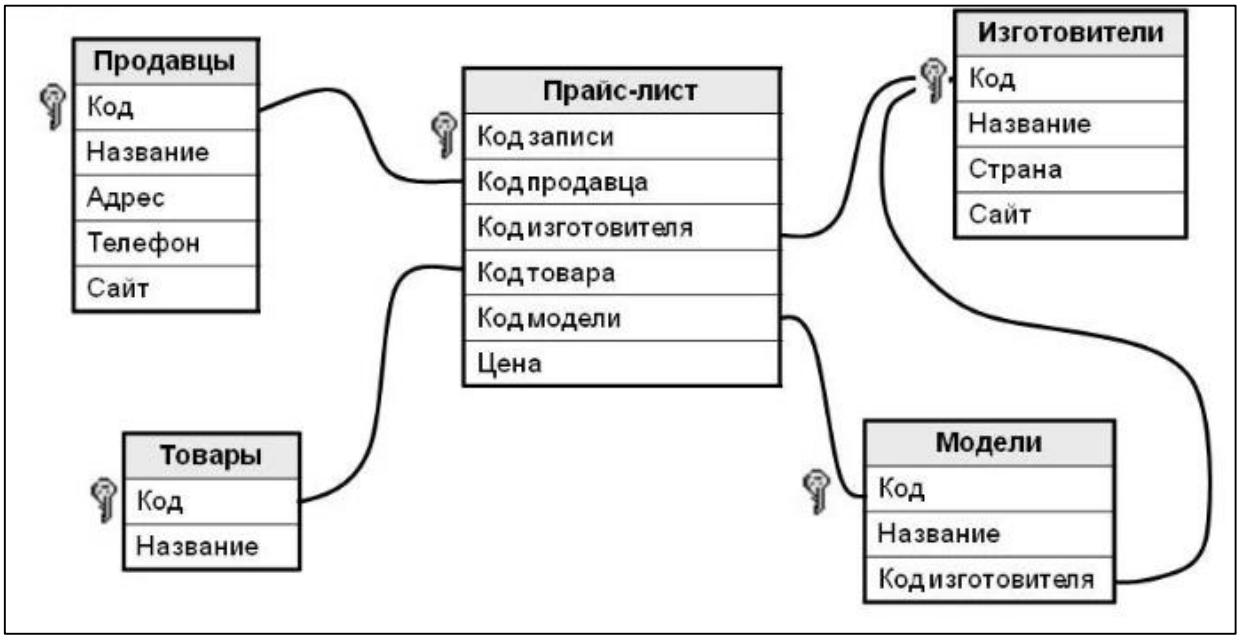

Рисунок 4 – Пример представления информации с помощью реляционной модели данных

Термин «реляционный» означает, что теория основана на математическом понятии отношение (relation). В качестве неформального синонима термину «отношение» часто встречается слово таблица.

Термин «таблица» не полностью соответствует понятию «отношение»: он лишь описывает визуальное представление отношений, которое более понятно человеку, тогда как термин «отношения» является общим понятием более высокого уровня абстракции. Тем не менее, при описании реляционных баз данных часто используется именно слово «таблица».

Существуют также и другие виды БД, но перечисленные выше – проверенные временем и наиболее часто используемые. Подробней реляционные базы данных и некоторые другие их виды будут рассмотрены далее.

1.2 Применение и значимость SQL

Согласно исследованиям аналитической компании Burning Glass (рисунок 5), SQL стал самым востребованным языком программирования в 2021 г, обойдя в рейтинге Python, Java и JavaScript.

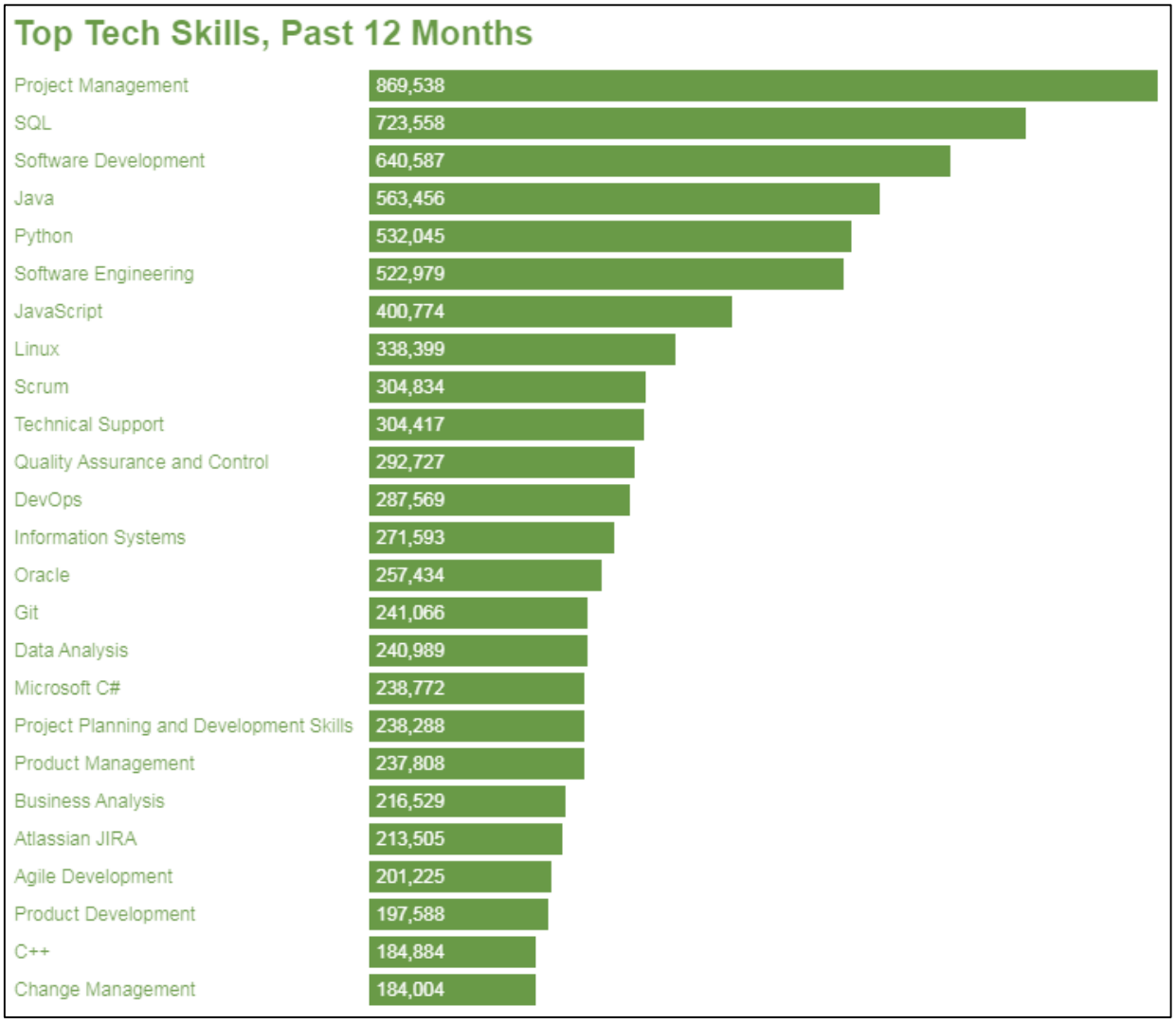

Рисунок 5 – Результаты исследований самых востребованных технических навыков по количеству вакансий аналитической компании Burning Glass за 2021 г.

SQL (англ. Structured Query Language – «язык структурированных запросов») – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Как видно из определения выше, SQL взаимодействует только с реляционными базами данных, что, тем не менее, не мешает ему по сей день оставаться одним из самых востребованных языков программирования, хоть он и появился в 1974 г.

Данный язык является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных.

Изначально SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

- создание в базе данных новой таблицы;
- добавление в таблицу новых записей;
- изменение записей;
- удаление записей;
- выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
- изменение структур таблиц.

Со временем SQL усложнился – обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) – и стал приобретать черты, свойственные большинству языков программирования.

При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

В реляционных хранилищах данных используются реляционные системы управления базами данных (РСУБД, Relational Database Management Systems, RDBMS).

На рисунке 6 приведены несколько наиболее известных РСУБД и указаны некоторые их характеристики (предоставляемый тип доступа для пользователей, языки программирования, используя которые данные РСУБД были разработаны, и примеры программных приложений и сервисов, использующих указанные РСУБД).

В РСУБД данные хранятся в табличном формате. Таблица – это базовая единица базы данных, она состоит из строк и столбцов, в которых хранятся данные, и представляет собой, как уже упоминалось в предыдущей части данной главы, «визуализированный» способ представления отношений в реляционных БД.

Система	Тип доступа	Написана на языке	Где используется
PostgreSQL	открытый	C	Skype, TripAdvisor, Яндекс.Почта, Avito
MySQL	открытый	C C++	GitHub, Facebook, Twitter, YouTube
SQLite	открытый	C	Adobe Photoshop Lightroom, Mozilla Firefox, Opera, Viber
Oracle	платный	Java C C++	МТС, Теле2, банк «Открытие», ВТБ
Google Cloud Spanner	общедоступный	C C++	PayPal, P&G, Lucille Games

Рисунок 6 – Примеры РСУБД

Таблицы – это самый часто используемый тип объектов баз данных, или структур, которые в реляционной БД хранят данные или ссылки на них. Среди других типов объектов баз данных можно отметить следующие:

- Представления (views) – виртуальные представления данных, собранных из одной или нескольких таблиц базы данных.
- Индексы (indexes) – таблицы, используемые для поиска данных, помогающие ускорить соответствующие возможности БД.
- Отчёты (reports) – они состоят из данных, полученных из одной или большего количества таблиц. Обычно они представляют некое подмножество данных, выбранных из БД на основе каких-то критериев поиска.

В РСУБД существуют следующие типы связей таблиц:

- Многие-ко-многим (many-to-many)

Эта связь позволяет нескольким записям в первой таблице ссылаться на несколько записей во второй, и наоборот (например, у одного учителя в школе может быть несколько уроков, а один урок могут вести несколько учителей). Такая связь часто не рекомендуется к использованию, а многие РСУБД и вовсе не позволяют ее реализовать. Альтернативой и хорошим тоном построения схемы базы данных считается организация такой связи с помощью вспомогательной таблицы.

- Многие-к-одному (many-to-one)

Эта связь позволяет нескольким записям в одной таблице ссылаться на одну и только одну запись в другой, а записи из другой таблицы при правильном построении структуры данных не содержат ссылок на первую таблицу вовсе (например, в состав одного класса входят несколько учеников, и каждый ученик может входить в состав лишь одного класса).

- Один-ко-многим (one-to-many)

Эта связь является обратной к предыдущей. В большинстве РСУБД две эти связи не разделяются теоретически и реализуются одинаково (т. е., например, только связь «один-ко-многим», но не «многие-к-одному»).

• Один-к-одному (one-to-one)

Эта связь позволяет записи из одной таблицы ссылаться только на одну запись в другой (например, у каждого ученика есть свой стул, и на каждом стуле сидит только один ученик).

SQL предоставляет механизм взаимодействия с объектами базы данных, который позволяет воспользоваться тем, что заложено в неё на этапе её проектирования.

Существуют различные подмножества команд SQL, в их состав, кроме прочих, входят следующие:

1. Язык описания данных (Data Definition Language, DDL) – это команды, которые ещё называют командами описания данных, так как они используются при описании структуры таблиц баз данных.

2. Язык управления данными (Data Manipulation Language, DML) – это команды, которые используются для управления данными в существующих таблицах путём добавления, изменения или удаления данных. В отличие от команд группы DDL, которые описывают то, как хранятся данные, DML-команды работают в уже существующих таблицах, описанных с помощью DDL-команд.

3. Язык запросов данных (Data Query Language, DQL) – эта группа состоит всего из одной команды SELECT, которая используется для получения необходимых данных из таблиц. Эту команду иногда включают в состав команд группы DML.

4. Язык для осуществления административных функций (Data Control Language, DCL) – команды из этой группы применяются для управления пользователями, для предоставления или отмены доступа к базе данных.

5. Язык для управления транзакциями (Transaction Control Language, TCL) – это команды, используемые для изменения состояния неких данных. Например – это команда COMMIT, заканчивающая текущую транзакцию с сохранением изменений в базе данных, или команда ROLLBACK, заканчивающая текущую транзакцию с отменой изменений в базе данных.

Стоит отметить, что в 2000-е годы стали стремительно развиваться т. н. базы данных NoSQL в качестве альтернативы реляционным БД и РСУБД, которые в то время уже активно использовались.

NoSQL – это сокращение от not only SQL (не только SQL). Этим термином обозначают базу данных любого типа, которая хранит информацию не так, как хранят её РСУБД. NoSQL-базы данных отлично подходят для многих современных приложений, которым нужны гибкие, масштабируемые, высокопроизводительные базы данных, обладающие широким набором функциональных возможностей. Среди таких приложений – проекты из мобильной и веб-сферы, компьютерные игры.

Причина роста популярности NoSQL-баз данных, в основном, кроется в необходимости работы с данными, которые имеют самую разную структуру и самые разные размеры. Это могут быть структурированные (structured), полуструктурированные (semi-structured) и полиморфные (polymorphic) данные. При работе с такими данными определить схему данных

практически невозможно. NoSQL, кроме того, даёт разработчику большую гибкость в тех случаях, когда ему нужно быстро адаптировать систему к изменениям. NoSQL-базы данных предлагают пользователям API с широкими функциональными возможностями, а также типы данных, спроектированные специально для определённых моделей данных. Считается также, что NoSQL-БД проще поддаются горизонтальному масштабированию.

Существует множество NoSQL-БД, но к основным относят следующие:

1. Базы данных типа «ключ-значение» (key-value database)

Это простейшая БД NoSQL, где у каждой записи есть ключ и значение, связанное с этим ключом. Подобные БД пригодятся в случаях, когда имеются большие наборы данных, по которым нужно быстро производить поиск. В таких базах данных управление ключами – это жизненно важная задача, так как каждое значение связано с уникальным ключом. В key-value БД все данные хранятся в одном и том же пространстве имён. Пример такой NoSQL-БД – Redis. Визуализированный пример key-value БД представлен на рисунке 7.

Key	Value
1	"apple"
2	1
3	{name: "Ann", surname: "Smith"}
4	null
5	"cherry"

Рисунок 7 – Пример key-value БД

2. Документоориентированные базы данных (document-oriented database)

Пример такой NoSQL-БД – Mongo DB. Визуализированный пример документоориентированной БД представлен на рисунке 8.

Content

{
name: "Ann",
surname: "Smith"
age: "35",
address: "24 Finsbury Street"
}

{
name: "Boris",
surname: "Smith"
age: "51",
address: "15 St Albans Street"
}

Рисунок 8 – Пример документоориентированной БД

Так называются базы данных, каждая запись которых представляет собой документ, содержащий поля и значения. Формат описания таких документов похож на JSON, XML или на двоичное представление формата JSON. Данные в таких БД организованы в коллекции (Collections), которые аналогичны таблицам в РСУБД. В одной коллекции обычно хранят данные, у которых есть что-то общее.

3. Колоночные базы данных (wide-column database)

В таких БД для хранения данных используются таблицы, строки и столбцы (колонки). Но, в отличие от реляционных баз данных, имена и форматы столбцов в одной и той же таблице могут отличаться. В колоночных NoSQL-базах данных чтение или запись «строки данных» состоит из чтения или записи отдельных столбцов. Столбец сохраняется в БД лишь тогда, когда в нём имеется элемент данных. К каждому элементу данных можно обратиться по ключу строки. При этом, выполнение запросов значений работает быстро (как запросы по индексу, а не как просмотр таблиц в РСУБД). Примеры колоночных БД – Cassandra и Clickhouse. Визуализированный пример колоночной БД представлен на рисунке 9.

Число верно выполненных заданий	0	1	2	3	4	5	6	7	8	9
Частота	1	1	1	2	5	6	8	7	5	4

Рисунок 9 – Пример колоночной БД

4. Графовые базы данных (graph database)

Пример графовой БД – Orient DB. Простой визуализированный пример графовой БД представлен на рисунке 10.

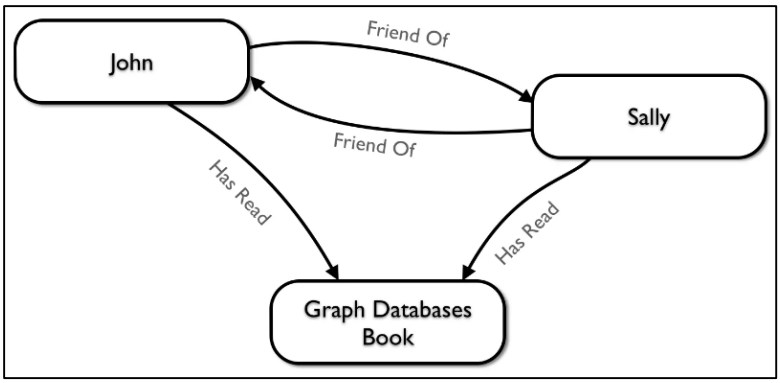

Рисунок 10 – Пример графовой БД

Это БД, которые хранят данные в узлах и рёбрах графов. Обычно данные о неких сущностях хранят в узлах, а сведения об отношениях между узлами хранят в рёбрах. Графовые базы данных созданы специально для того, чтобы хранить сведения о взаимоотношениях между сущностями и работать с этими сведениями. Отношения – это центральный элемент графовых баз данных. Ребра характеризуются начальным узлом, обозначающим сущность,

конечным узлом, типом отношений, направлением. Используются такие БД для анализа и просмотра отношений между связанными данными (обнаружение мошенничеств, рекомендательные системы).

NoSQL-БД действительно востребованы, однако вытеснить SQL им так и не удалось. Это связано с тем, что SQL-базы данных прошли долгий путь развития, и вокруг этой идеологии работы с данными сложилось целое сообщество. РСУБД в настоящее время предоставляют намного больший функционал, чем на ранних этапах развития, и, в целом, реляционные БД предназначены для решения широкого круга задач. Тогда как NoSQL-БД, будучи более гибкими, чем достаточно строго унифицированные SQL-БД, тем не менее, как правило, более узкоориентированы. К тому же, со временем некоторые из NoSQL-СУБД обзавелись специфическими SQL-подобными языками запросов (CQL, N1QL, AQL и др.), что еще раз подтверждает актуальность изучения SQL на данный момент.

На рубеже 2000-х и 2010-х годов появились т. н. NewSQL-БД (например, VoltDB) как ответ на потребности рынка, которые существующие БД не могли удовлетворить. Так, SQL-базы не поддерживали масштабируемость на уровне NoSQL, а эти, в свою очередь, не отвечали стандартам точного выполнения оперативных транзакций ACID (англ. atomicity, consistency, isolation, durability – «атомарность, непротиворечивость, изолированность, долговечность»).

NewSQL совмещают реляционную модель, язык запросов SQL и распределённые горизонтально масштабируемые базы данных NoSQL.

Для баз данных NewSQL характерны:

- реляционная модель и транзакционность;
- язык SQL для доступа к данным;
- горизонтальная масштабируемость;
- более быстрая производительность за счёт новых «движков».

Как указано выше, NewSQL-БД также используют SQL.

1.3 Роль мобильных приложений в современности

В современном мире уже не говорят, что за мобильными устройствами будущее – мы уже в нем живем.

Скорость роста рынка мобильных приложений не собирается снижаться или стабилизироваться. Пандемия оказала свое влияние на тенденции среди предпочтений пользователей: 26 июля 2021 года Cisco AppDynamics поделилась итогами отчёта App Attention Index, который продемонстрировал, что зависимость потребителей от приложений и цифровых сервисов резко возросла с начала пандемии COVID-19. Кроме этого, смена парадигмы рабочего пространства, удаленный рабочий процесс, ограничения в сфере развлечений – все это и многое другое продолжает формировать постоянный спрос и провоцировать дальнейший рост мобильного сегмента рынка.

Согласно исследованиям Mary Meeker, люди все больше пользуются мобильным интернетом: с каждым годом увеличивается разрыв между трафиком с десктопных и мобильных устройств (Рисунок 11). Например, в

2018 году – в среднем 3.6 часов в мобильном (что уже в 12 раз больше, чем показатель 2008 года) и всего 2 часа в день с компьютера (этот показатель стабилен).

Рисунок 11 – Диаграмма, показывающая рост использования мобильного интернета по сравнению с десктопным

С колоссальной скоростью растет и количество предлагаемых к использованию мобильных приложений (рисунок 12 на примере Google Play (Statista)).

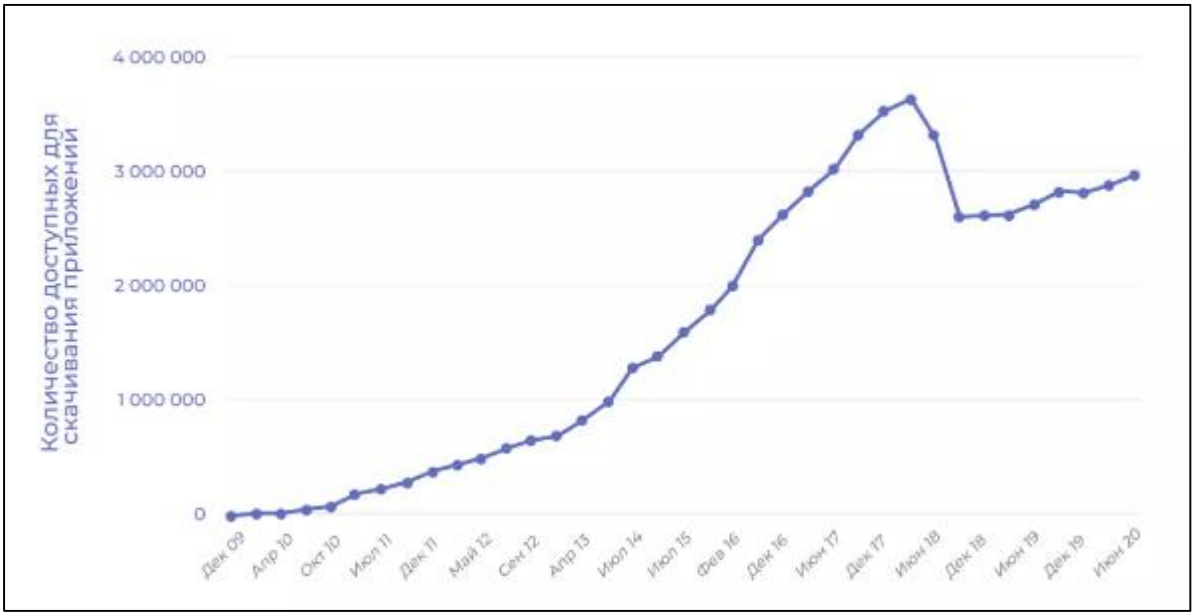

Рисунок 12 – Диаграмма, показывающая рост количества доступных для скачивания мобильных приложений

Проанализировав динамику появления новых мобильных приложений в Google Play, можно заметить, что, в среднем, каждый месяц появляется более 100 тысяч новых приложений (рисунок 13 (Statista)).

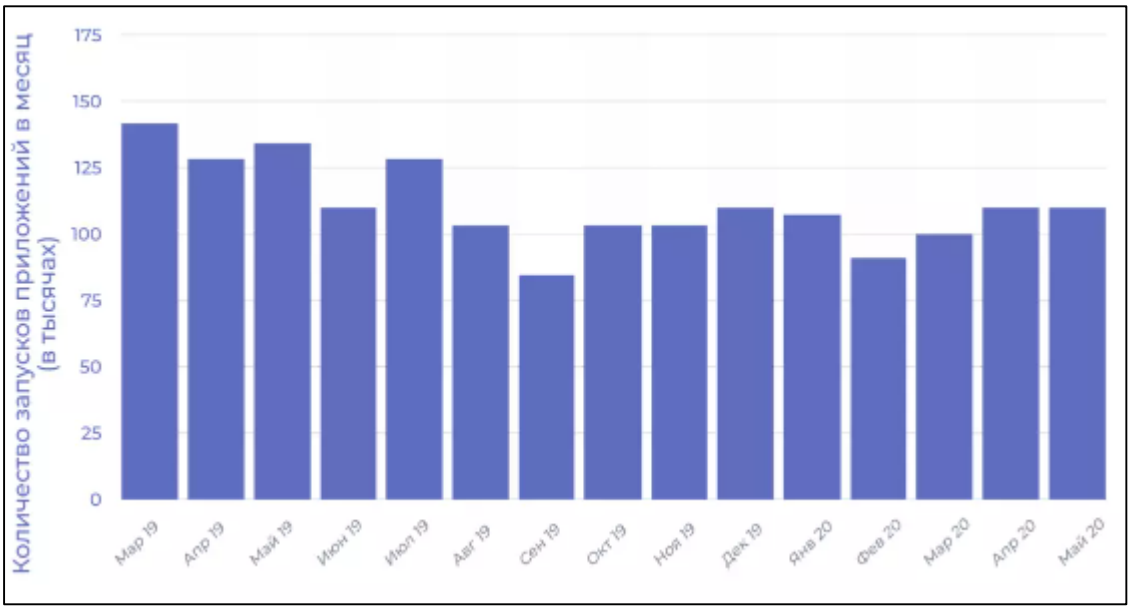

Рисунок 13 – Диаграмма, показывающая количество загружаемых в Google Play новых мобильных приложений в разрезе месяцев

По данным Statista на начало 2020 г. (рисунок 14), почти три четверти респондентов заявили, что на их телефонах установлено не менее 11-ти приложений (без учета предустановленных). Несмотря на то что тенденция загрузок новых программ стабилизируется, пользователи все еще готовы скачивать новинки.

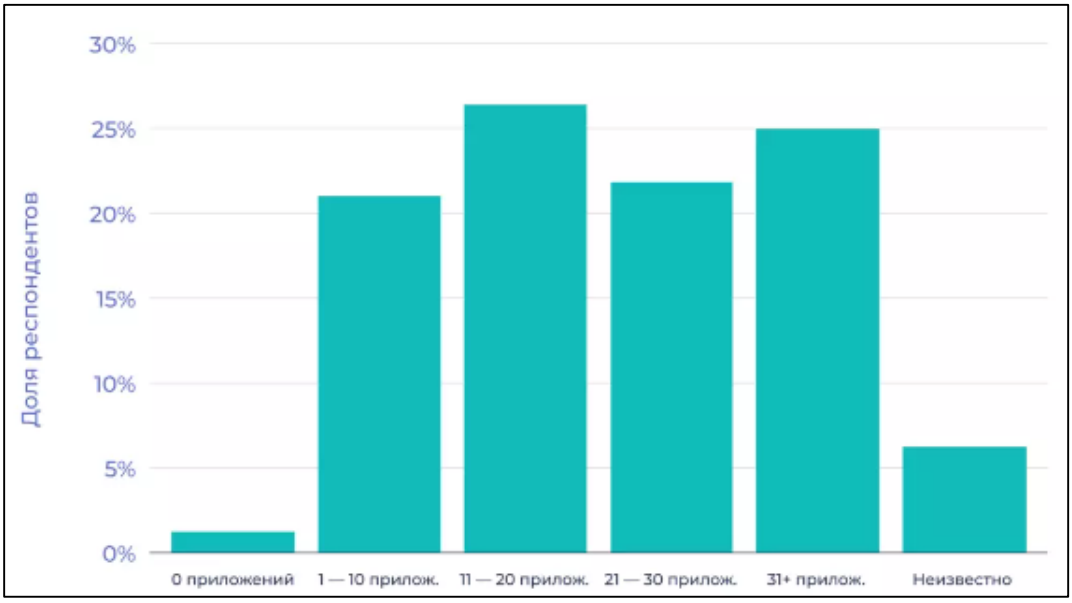

Рисунок 14 – Диаграмма, демонстрирующая соответствие между долей респондентов и количеством установленных приложений на их мобильных устройствах

Также стоит отметить, что согласно данным Counterpoin Research:

• Растет не только количество загрузок и потраченных денег, но и среднее время использования приложения;

• Мировой рынок смартфонов за 2021 г. вырос на 5,7%, до 1,35 млрд штук;

• Каждый четвертый пользователь смартфона проводит с ним более 7 часов.

Эти данные подтверждают, что мобильные приложения в современном обществе уже стали неотъемлемой часть повседневной жизни благодаря удобству в использовании и доступностью.

Мобильные приложения все чаще рассматриваются и как будущее обучения. Популярность мобильных устройств и широкое распространение mLearning или мобильного обучения набирает обороты.

Мобильные приложения в сфере образования и развития – это формат предоставления обучения, предназначенный для просмотра учебных материалов учащимися на мобильных устройствах в автономном режиме (как через Интернет, так и офлайн).

Люди все чаще обращаются к мобильным приложениям, и ежегодное количество их загрузок это подтверждает. Согласно статистики, число мобильных обучающих приложений, которые загружены по всему миру, превышает 2,6 миллиона.

Мобильные приложения – одна из самых популярных тенденций мобильного обучения, представленная сегодня на рынке. Этот формат имеет множество преимуществ:

• Идеально подходит для людей, которые ищут информацию и обучаются в дороге и путешествиях;

• Подходит как для просмотра онлайн, так и для офлайн;

• Очень популярен среди современного поколения;

• Способствует более высоким показателям завершения обучения;

• Обеспечивает доступ к оперативной информации;

• Идеально подходит для поддержки эффективности и производительности.

Мобильные приложения хорошо подходят как для формального, так и для неформального обучения. Их можно использовать, чтобы предложить небольшой фрагмент формального обучения или дополнить уже существующее обучение.

Когда дело доходит до неформального обучения, мобильные приложения являются идеальной платформой для поддержки эффективности. Их можно использовать в качестве инструментов поддержки производительности, и они могут быть встроены в рабочий процесс учащихся. Если они будут доступны ученикам на их мобильных устройствах, то они смогут использовать их для получения своевременной информации и поддержки на рабочем месте.

Существует множество способов интеграции мобильных приложений для обучения в учебную стратегию:

1. Для вводного обучения (формальное обучение)

Мобильные приложения могут быть использованы для формального обучения в форме коротких чек-листов, которые могут быть частью траектории обучения.

2. Как дополнение к формальному обучению

- Они могут быть использованы в качестве предварительной и последующей оценки для формального обучения (онлайн или смешанного)
- Также могут быть использованы для демонстрации видео, примеров и сценариев для усиления обучения

Можно выделить следующие причины, почему мобильные приложения подходят для организации стратегии обучения:

- Гибкость использования

Мобильные приложения стали неотъемлемой частью нашей жизни благодаря гибкости и простоте поиска информации, которую они предлагают. Возможности мобильных приложений могут быть использованы для обучения учеников, даже если они не подключены к Интернету.

- Высокие показатели завершения обучения

Возможность обучения в любое время и в любом месте, которую предлагают мобильные обучающие технологии и мобильные приложения, помогает учащимся учиться, когда они «хотят», а не «должны», что приводит к более высоким показателям завершения обучения.

- Вовлечение

При использовании смартфона почти все, что делает пользователь: отправка сообщений, поиск кафе поблизости, просмотр ежедневных новостей и т. д. – осуществляется через приложения. Например, многие отказываются от традиционных SMS-сообщений в пользу таких приложений, как WhatsApp, Viber и др. по той простой причине, что они гораздо более живые и привлекательные и имеют множество функций.

Форматы мобильных приложений и учебные мобильные технологии заметно отличаются от традиционного электронного обучения. Они обеспечивают очень высокую вовлеченность учащихся, тем самым повышая эффективность запоминания и удержания учебного материала.

- Ориентация на современное поколение

Когда дело касается современного молодого поколения, количество часов, потраченных на мобильные приложения в сравнении с более взрослым поколением возрастает значительно.

- Простота обновлений

Мобильные приложения чрезвычайно удобны в использовании и предлагают организациям гибкость, которая позволяет легко обновить их при необходимости.

- Возможность использования современных подходов в обучении

Можно использовать такие тенденции, как микрообучение, социальное обучение и геймификация.

- Могут быть адаптированы к различным потребностям обучения

Создавать мобильные приложения для обучения можно так, чтобы удовлетворить различные учебные потребности, включая обучение соответствию требованиям, развитие мягких навыков и т.д.

• Могут использоваться для поддержки эффективности и производительности

Мобильные приложения предназначены для предоставления информации «точно в срок». Они идеально подходят, чтобы обеспечить поддержку эффективности, и могут непосредственно влиять на использование приобретенных знаний на рабочем месте.

• Могут использоваться как дополнение к смешанному или формальному обучению

Использовать формальное или смешанное обучение можно с помощью инструкций, чек-листов или других инструментов поддержки эффективности. Но дополнение имеющегося обучения мобильными приложениями позволит улучшить применение полученных знаний непосредственно в работе.

1.4 Анализ существующих аналогов

Стоит отметить, что, в связи с востребованностью и популярностью SQL, а также наличием в настоящее время значительного количества обучающих платформ, предоставляющих различные образовательные курсы, в том числе и в формате мобильного приложения, существует достаточно много аналогов для описываемого проекта (в том числе и достаточно популярные образовательные платформы).

Рассмотрим несколько из них:

• Курс «Интерактивный тренажер по SQL» на образовательной платформе Stepik

Stepik (Стэпик, до августа 2016 года – Stepic) – российская образовательная платформа и конструктор бесплатных и платных открытых онлайн-курсов и уроков (рисунок 15).

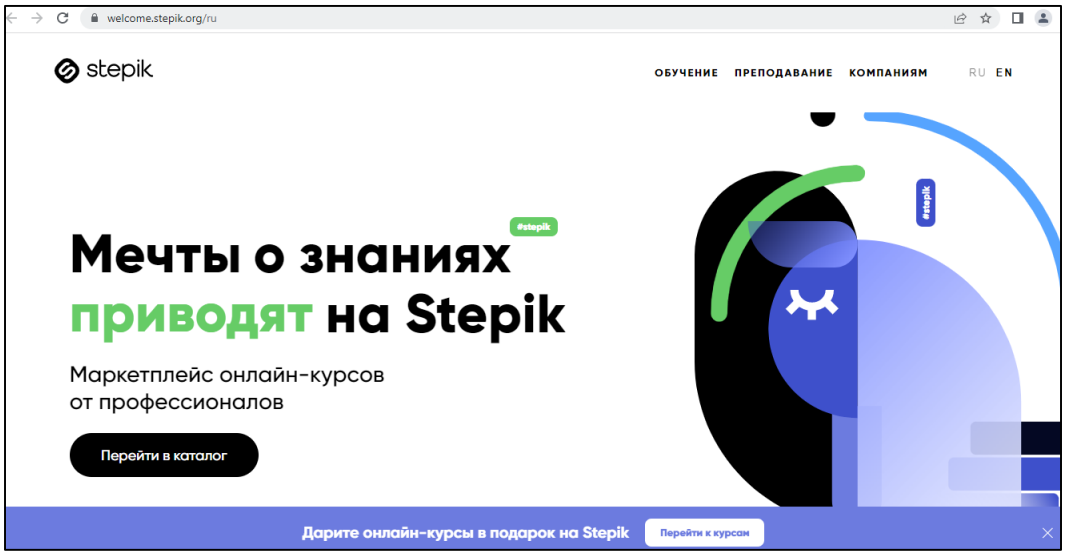

Рисунок 15 – Главная страница сайта образовательной платформы «Stepik»

Платформа позволяет любому зарегистрированному пользователю создавать интерактивные обучающие уроки и онлайн-курсы, используя видео, тексты и разнообразные задачи с автоматической проверкой и моментальной обратной связью. В процессе обучения студенты могут вести обсуждения между собой и задавать вопросы преподавателю на форуме. Основные охватываемые курсами дисциплины – программирование, математика, биоинформатика и биология, экономика; основной язык курсов – русский, есть курсы на английском языке. По состоянию на 2020 год на платформе зарегистрировано 5 миллионов пользователей

Мобильное приложение данной образовательной платформы можно бесплатно скачать в Google Play.

Данная платформа на запрос «SQL» предлагает лишь один бесплатный курс: «Интерактивный тренажер по SQL» (рисунок 16).

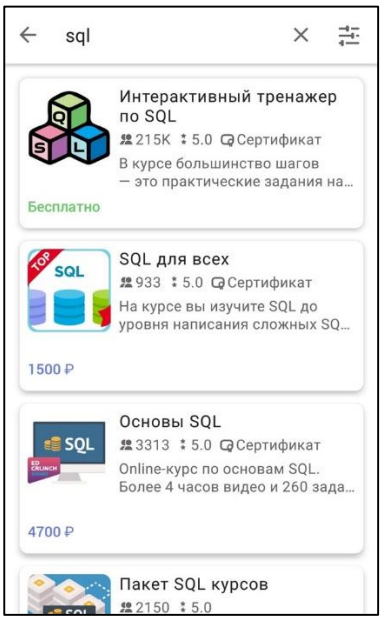

Рисунок 16 – Курсы по SQL на платформе Stepik

В описании к данному курсу помимо информации, представленной на рисунке 17, указано: «В курсе большинство шагов – это практические задания на создание SQL-запросов. Каждый шаг включает минимальные теоретические аспекты по базам данных или языку SQL, примеры похожих запросов и пояснение к реализации.

Для создания, выполнения и отладки SQL-запросов используется платформа Stepik, на свой компьютер ничего дополнительно устанавливать не надо.

Сложность запросов возрастает по мере прохождения курса. Сначала они формулируются для отдельных таблиц, а затем для баз данных, разработанных для предметных областей, таких как "Интернет-магазин", "Тестирование", "Абитуриент". Причем в процессе выполнения шагов курса решаются практические задачи из выбранной предметной области.

Каждый учащийся может придумать свои задания на создание SQL-запросов. В курсе есть модуль, в котором размещаются лучшие из них.

Данный курс направлен на то, чтобы научить слушателя создавать базы данных и реализовывать запросы к ним на языке SQL для различных предметных областей.

Кому подойдет данный курс?

- начинающим аналитикам;
- разработчикам;
- маркетологам».

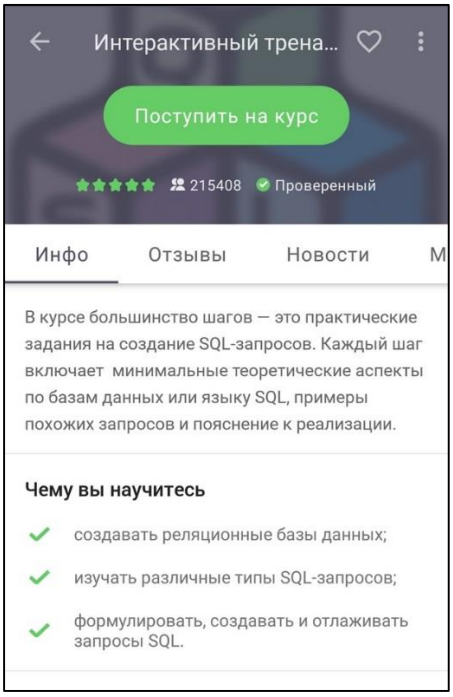

Рисунок 17 – Некоторая информация о курсе «Интерактивный тренажер по SQL»

Программа курса следующая:

1. Основы реляционной модели и SQL
- Отношение (таблица)
- Выборка данных
- Запросы, групповые операции
- Вложенные запросы
- Запросы корректировки данных
- Таблица "Командировки", запросы на выборку
- Таблица "Нарушения ПДД", запросы корректировки
- Глоссарий и поиск по курсу

2. Запросы SQL к связанным таблицам
- Связи между таблицами
- Запросы на выборку, соединение таблиц
- Запросы корректировки, соединение таблиц
- База данных «Интернет-магазин книг», запросы на выборку
- База данных «Интернет-магазин книг», запросы корректировки

3. Базы данных и SQL запросы
- База данных «Тестирование», запросы на выборку

- База данных «Тестирование», запросы корректировки
- База данных «Абитуриент», запросы на выборку
- База данных «Абитуриент», запросы корректировки
- База данных "Учебная аналитика по курсу"

4. SQL запросы пользователей

- База данных «Интернет-магазин книг», часть 1
- База данных «Интернет-магазин книг», часть 2
- База данных «Интернет-магазин книг», часть 3
- База данных «Тестирование»

Каждый урок на курсе разбивается на «шаги». На платформе Stepik имеется два типа таких шагов:

1) предоставление теоретической информации (рисунок 18);

2) задание для контроля усвоения информации (Рисунки 19 и 20).

Рисунок 18 – Блок теории на курсе от Stepik

Рисунок 19 – Блок заданий на курсе от Stepik

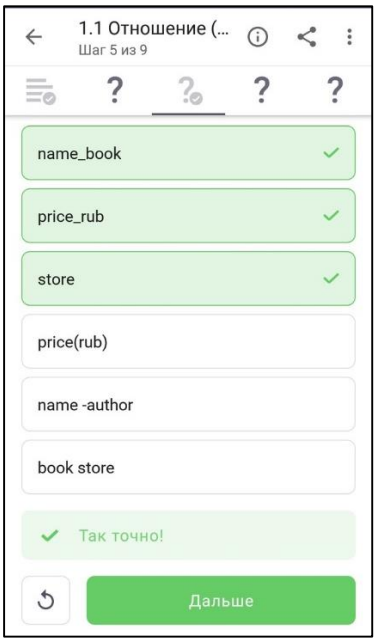

Рисунок 20 – Блок заданий на курсе от Stepik

Мобильное приложение образовательной платформы Stepik (как видно изприведенных выше скриншотов) удобно в использовании, оно имеет интуитивно понятный и приятный для работы интерфейс.

- Курс «SQL» на образовательной платформе SoloLearn

Образовательная платформа SoloLearn (рисунок 21) ориентирована на всех тех, кто хочет связать свою жизнь с IT или просто интересуется программированием. При этом функционирование платформы больше ориентировано на использование мобильного приложения, а не десктопной версии.

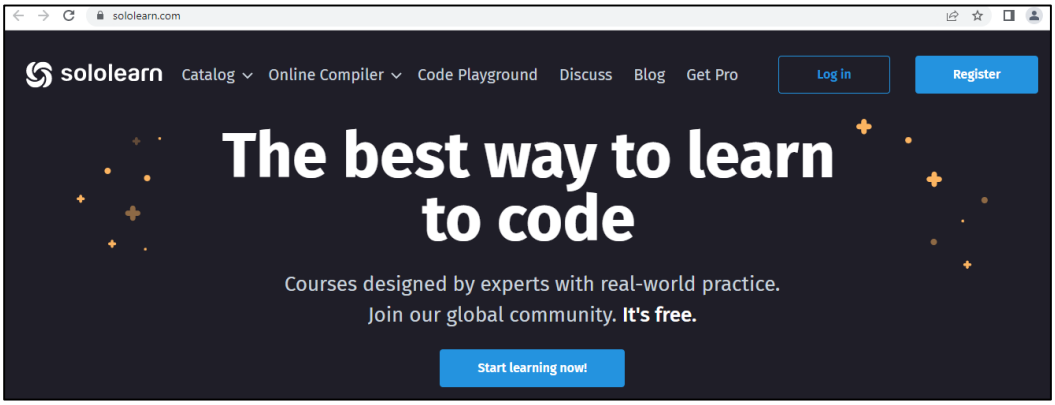

Рисунок 21 – Главная страница сайта образовательной платформы «SoloLearn»

Образовательная платформа SoloLearn предоставляет для изучения достаточно большое количество курсов по известным и популярным языкам программирования. Среди них есть и курс для изучения SQL (рисунок 22 и рисунок 23).

Образовательный процесс также разбит на шаги (см. рисунок 23).

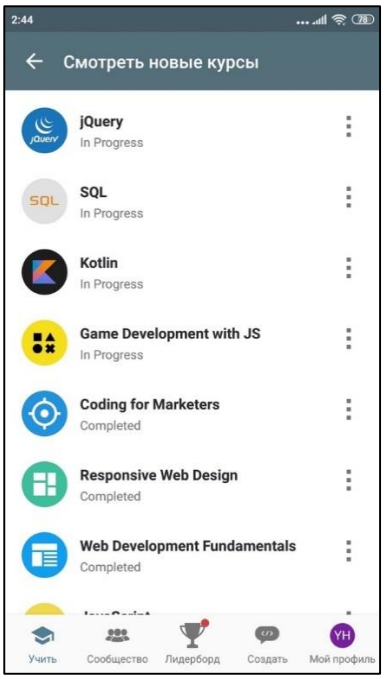

Рисунок 22 – Некоторые из курсов, предоставляемые на образовательной платформе «SoloLearn»

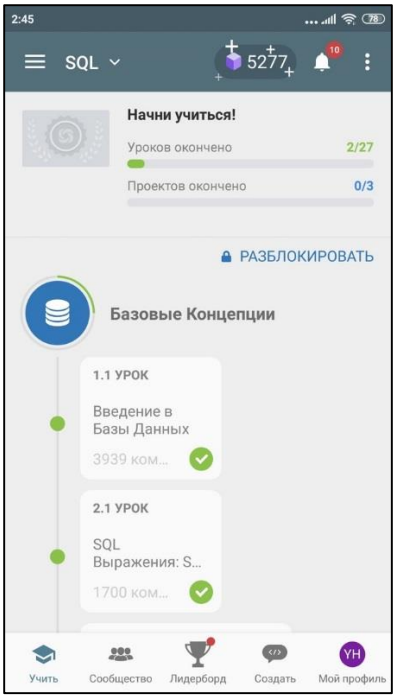

Рисунок 23 – Первые этапы изучения курса по SQL на SoloLearn

Обучение на курсах SoloLearn организовано таким образом, чтобы блоки теории и примеров (рисунок 24) чередовались в своей последовательности с блоками заданий (рисунок 25) (т. е. чтоб после каждого теоретического шага следовало соответствующее задание).

Интерфейс мобильного приложения образовательной платформы «Sololearn» также интуитивно понятен и прост в использовании

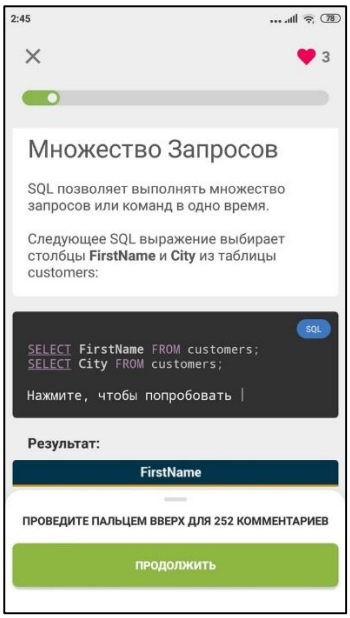

Рисунок 24 – Блок теории на курсе от SoloLearn

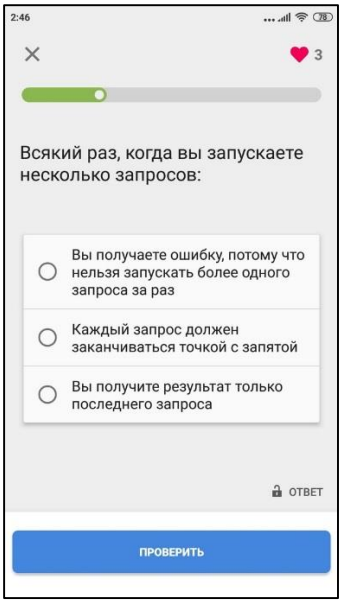

Рисунок 25 – Блок заданий на курсе от SoloLearn

Основная проблема рассмотренных обучающих курсов по SQL (как и многих других) заключается в том, что предоставляемая бесплатно информация достаточно скудна и поверхностна: зачастую на таких курсах изучается лишь основополагающая базовая информация. Так, например, при изучении SQL часто ограничиваются лишь несколькими составляющими (например, DDL и DQL) или программа обучения включает только информацию о синтаксисе SQL, но не затрагивает теория баз данных, и т.д.

2 ПРОЕКТИРОВАНИЕ ТРЕБОВАНИЙ

2.1 Функциональные требования

Функциональные требования определяют функциональность ПО, которую нужно построить, чтобы пользователи смогли выполнить свои задачи (т.е. что должно или не должно делать приложение или отдельные его компоненты).

К мобильному приложению для изучения SQL были выявлены следующие функциональные требования:

1. Мобильное приложение должно предоставлять пользователю доступ к последовательности уроков;
2. Мобильное приложение должно отображать пользователю теоретический материал урока при переходе пользователя на теоретический этап занятия;
3. Мобильное приложение должно отображать пользователю условия тестовых заданий по пройденной пользователем теоретической части после перехода последнего на тестовый этап занятия;
4. Мобильное приложение должно принимать и обрабатывать выбранный пользователем ответ на тестовое задание;
5. Мобильное приложение должно выводить сообщение об ошибке при неверно выбранном варианте ответа на тестовое задание и сообщение о правильности выбранного ответа в случае его корректности с последующим переходом к следующему заданию;
6. Мобильное приложение должно сохранять тестовые задания, на которые пользователь дал неверный ответ, и предоставлять пользователю возможность пройти данные задания еще раз;
7. После того, как пользователь закончил тестовые задания по уроку, мобильное приложение должно осуществлять переход к этапу заданий по коду SQL;
8. Мобильное приложение должно отображать пользователю условия заданий по коду SQL и предоставлять возможность ввода ответа;
9. Мобильное приложение должно принимать ответ пользователя на задание по коду SQL и обрабатывать его;
10. Мобильное приложение должно выводить сообщение об ошибке при неверном ответе на задание по коду SQL и предлагать возможность просмотреть решение с пояснениями и сообщение о правильности выбранного ответа в случае его корректности с последующим переходом к следующему заданию;
11. Мобильное приложение должно сохранять задания по коду SQL, на которые пользователь дал неверный ответ, и предоставлять пользователю возможность пройти данные задания еще раз;
12. Мобильное приложение должно сохранять пользовательский прогресс по курсу;
13. Мобильное приложение должно предоставлять возможность просмотреть изученные уроки;

14. Мобильное приложение должно предоставлять возможность повторить материал пройденных уроков путем повторного прохождения пользователем заданий по соответствующей теме;

15. Мобильное приложение должно запрещать пользователю доступ к заблокированным урокам;

16. Мобильное приложение должно предоставлять возможность пройти тест по случайным заданиям по пройденному материалу;

17. Мобильное приложение должно сохранять статистику по пройденному пользователем этапу (количество пройденных всего тестовых заданий, количество пройденных всего заданий по коду, количество неверных тестовых заданий, количество неверных заданий по коду и т.п.);

18. Мобильное приложение должно предоставлять пользователю информацию о текущем прогрессе (доля пройденного материала, доля верно отвеченных заданий и т.п.)

19. В программе занятий должна быть информация не только о синтаксисе языка SQL, но также и об основах теории баз данных и СУБД;

20. При использовании нестандартных функций языка SQL (спецификаций конкретной СУБД) должна присутствовать соответствующая сноска;

21. В мобильном приложении должен вестись глоссарий терминов по пройденным пользователем урокам;

21. Мобильное приложение должно предоставлять пользователю возможность воспользоваться глоссарием по пройденным урокам.

2.2 Нефункциональные требования

Нефункциональные требования определяют стандарты производительности и атрибуты качества программного обеспечения, например, удобство использования системы, эффективность, безопасность, масштабируемость и т.д.

Если приложение не соответствует нефункциональным требованиям, оно продолжает выполнять свои основные функции, однако не сможет обеспечить удобство для пользователя.

К мобильному приложению для изучения SQL были выявлены следующие нефункциональные требования:

1. Загрузка мобильного приложения не должна занимать более 20 секунд;

2. В случае задержки загрузки приложения необходимо выводить экран загрузки;

3. Мобильное приложение должно обеспечивать масштабируемость (приложение должно корректно функционировать при изменении масштаба содержимого или других параметрах экрана);

4. В мобильном приложении должен быть предусмотрен раздел «Помощь», в котором пользователю предлагается возможность узнать о приложении в целом и/или об отдельных ее компонентах (экраны, кнопки, связи и т.п.);

5. Мобильное приложение не должно взаимодействовать, хранить, требовать доступ к личным данным пользователя (галерея, камера, микрофон и т.п.);

6. Интерфейс приложения не должен содержать ярких едких цветов, предпочтительно использование сочетающегося и неброского оформления (дизайнерская составляющая не должна отвлекать пользователяот обучающего контента);

7. Мобильное приложение должно работать на платформе Android версии 4.0 и выше;

8. Мобильное приложение должно быть написано на языке Java с использованием Android SDK (с использованием интегрированной среды разработки в свободном доступе Android Studio).

3 РАЗРАБОТКА АРХИТЕКТУРЫ

3.1 Диаграмма вариантов использования

Диаграмма вариантов использования (диаграмма прецедентов, UCD от англ. use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

Другими словами, это графическое изображение взаимодействий между элементами системы.

Данная диаграмма является диаграммой UML поведенческого типа.

Она часто используется для анализа различных систем, т. к. диаграмма вариантов использования позволяет визуализировать разные типы ролей в системе и как эти роли взаимодействуют с системой.

Следует отметить, что диаграмма использования не отражает детализацию вариантов использования, она лишь объединяет некоторые отношения между вариантами использования, актерами и системами.

При моделировании системы с помощью диаграммы прецедентов системный аналитик стремится:

- чётко отделить систему от её окружения;
- определить действующих лиц (актеров), их взаимодействие с системой и ожидаемую функциональность системы;
- определить в глоссарии предметной области понятия, относящиеся к детальному описанию функциональности системы (то есть прецедентов).

Работа над диаграммой может начаться с текстового описания, полученного при работе с заказчиком. При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов опускаются

Диаграмма вариантов использования определяет функциональное назначение моделируемой системы или предметной области

Основными элементами диаграммы вариантов использования являются актер и вариант использования.

Актер – это внешняя по отношению к моделируемой системе сущность, взаимодействующая с системой для решения некоторых задач. В качестве актера может использоваться человек, другая система, устройство или программное средство.

Вариант использования определяет некоторый набор действий (операций), которые должны быть выполнены моделируемой системой или программным средством при взаимодействии с актером.

На рисунке 26 представлена диаграмма вариантов использования для описываемого приложения.

На диаграмме выделено три роли:

- Непосредственно пользователь;
- База данных, в которой хранится информация о курсе и составе информационной части приложения;

• Программный модуль, который отвечает за непосредственную программную обработку данных в приложении (расчет статистики, сверка ответов и т.п.).

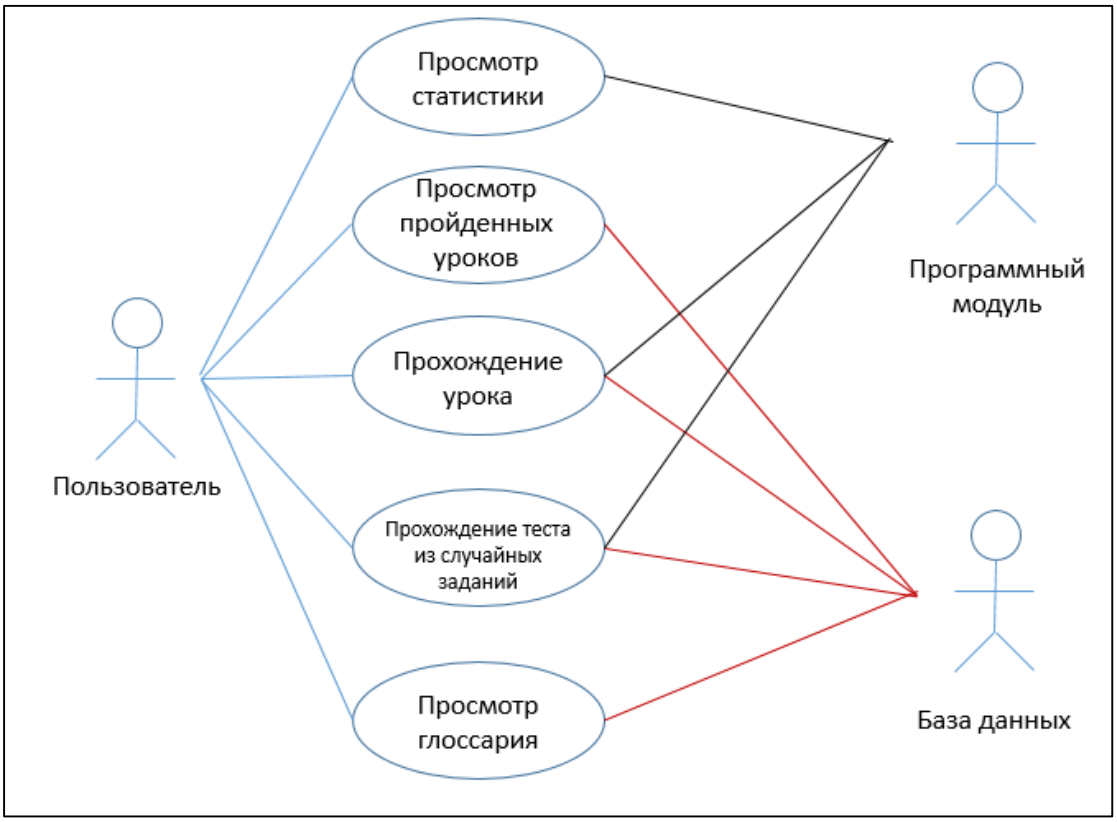

Рисунок 26 – Диаграмма вариантов использования для приложения

3.2 Структура данных приложения

Опишем данные, с которыми работает приложение.

Объект «Урок» помимо идентификатора и порядкового номера в курсе хранит в себе название темы, теорию и поле-перечисление, указывающее на состояние объекта (прошедший, текущий, не пройденный).

Таблица 1 – Структура объекта «Урок»

Урок	Описание
id	Идентификатор
position	Номер урока на курсе
theme	Название темы
theory	Собственно теория по уроку
status	Состояние объекта

Тестовое задание и задание по коду SQL также являются относительно независимыми элементами, которые в дальнейшем могут использоваться в приложении независимо от урока (например, при решении случайного задания, вопрос предоставляется пользователю без указания, к какому уроку он относится), поэтому выделим их в отдельные объекты, которые будут связаны с объектом урока по идентификатору.

Таблица 2 – Структура объекта «Тестовое задание»

Название поля	Описание
id	Идентификатор
lesson_id	Идентификатор урока, по теме которого поставлен вопрос
question	Условие задания (вопрос)
answer	Верный ответ
wrong_variants	Строка, в которой в определенном формате хранятся неправильные варианты ответа

Таблица 3 – Структура объекта «Задание по коду SQL»

Название поля	Описание
id	Идентификатор
lesson_id	Идентификатор урока, по теме которого поставлен вопрос
question	Условие задания (вопрос)
answer	Верный ответ

Отдельным объектом данных является «Термин» – именно из этих объектов формируется глоссарий.

Таблица 4 – Структура объекта «Термин»

Название поля	Описание
id	Идентификатор
lesson_id	Идентификатор урока, к теме которого относится термин
name	Сам термин
definition	Определение термина

В итоге, общая схема данных для приложения с указанием связей (все связи типа «один-ко-многим») представлена на рисунке 27.

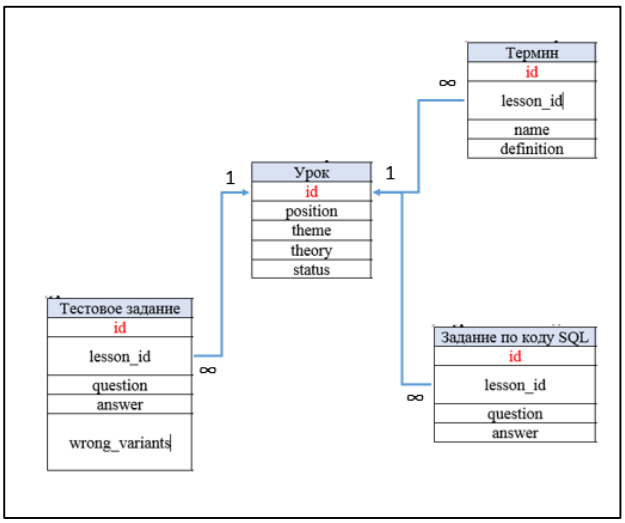

Рисунок 27 – Схема данных для мобильного приложения

3.3 Диаграмма последовательности

Для описания динамического аспекта системы, т. е. функционирования процессов во времени, используют такие UML-диаграммы, как диаграммы последовательности.

На рисунке 28 представлена диаграмма последовательности для одного из процессов, необходимость наличия которого указана в функциональных требованиях, и который может вызвать некоторые сложности в понимании.

На диаграмме выделены следующие роли:

- Пользователь

Тот, кто скачал и непосредственно взаимодействует с мобильным приложением.

- Интерфейс

Это не только графическое представление элементов экрана, но и некоторый код (обработчики событий наведения, например).

- Программный модуль

Это программный код, который непосредственно реализует бизнес-логику приложения.

- База данных

В ней хранится информация о курсе и составе информационной части приложения.

Отображение динамичной составляющей указанного процесса (последовательности выполнения его функций) представлено на рисунке 28.

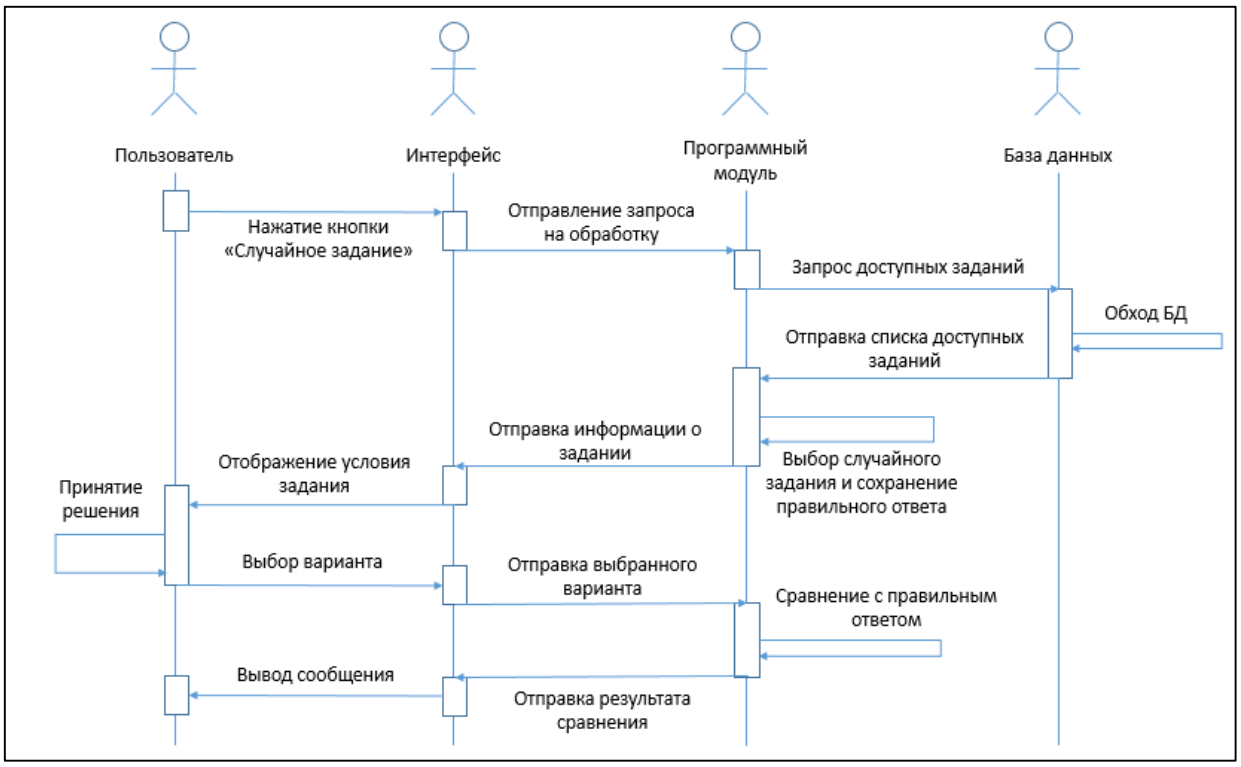

Рисунок 28 – Диаграмма последовательности для процесса прохождения случайного задания

4 ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА

4.1 Проектирование экранов

Всего в приложении должны быть реализованы следующие экраны:

1. Начальный экран (рисунок29)

От этого экрана можно перейти к любому другому, но основная его функция – отображение последовательности уроков в курсе.

2. Экран просмотра урока (рисунок 30)

Данный экран предоставляет возможность просмотреть непосредственно урок;

3. Экран просмотра статистики

Здесь можно увидеть прогресс по курсу, долю неверных ответов от общих (переход к этому экрану осуществляется путем нажатия на соответствующую иконку в верхнем левом углу экрана).

4. Экран просмотра глоссария

Здесь перечислены все термины и их определения по темам, которые пользователь уже прошел.

5. Экран тестового задания

Здесь пользователю выводится тестовое задание и предлагаются варианты ответа.

6. Экран задания по коду SQL

Аналогичен экрану тестового задания, но ответ пользователь не выбирает, а вводит сам.

Рисунок 29 – Пример дизайна начального экрана

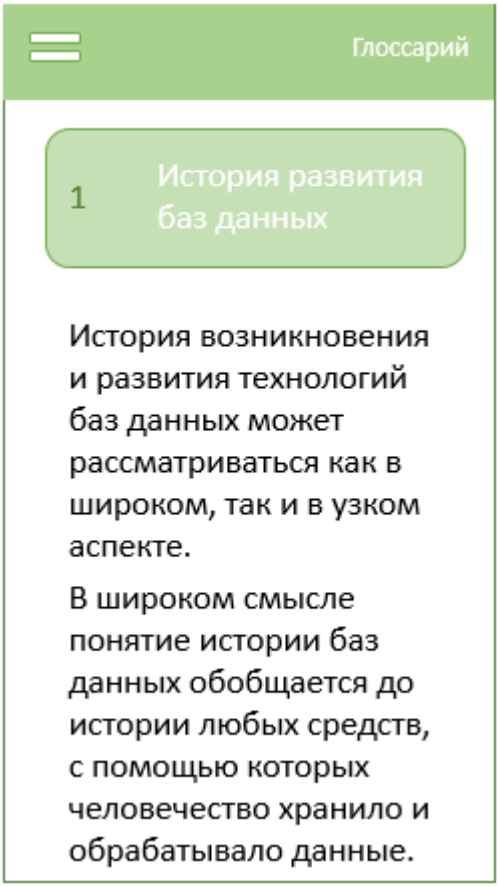

Рисунок 30 – Пример дизайна экрана просмотра урока

4.2 Взаимосвязь экранов

Зная, какие экраны имеется в приложении, можно построить схему взаимосвязи экранов приложения.

Для описываемого мобильного приложения такая схема приведена на рисунке 31.

Данные схемы позволяют установить все возможные переходы от одного экрана к другому. Это позволяет отследить возможную «траекторию движения» пользователя по приложению.

Кроме того, такая схема может быть весьма полезна для разработчиков, т. к. она предоставляет им возможность отследить все существующие переходы и, соответственно, предусмотреть всех их при программировании (т. е. реализовать).

На схеме связи экранов приложения можно увидеть, что начальный экран связан со всеми другими экранами.

С экранами заданий он связан с помощью возможность решить случайное задание.

Итого всего на схеме указано 7 связей, 5 из которых отходят от начального экрана.

Все экраны также предлагают возможность вернуть к начальном (см. рисунок 30).

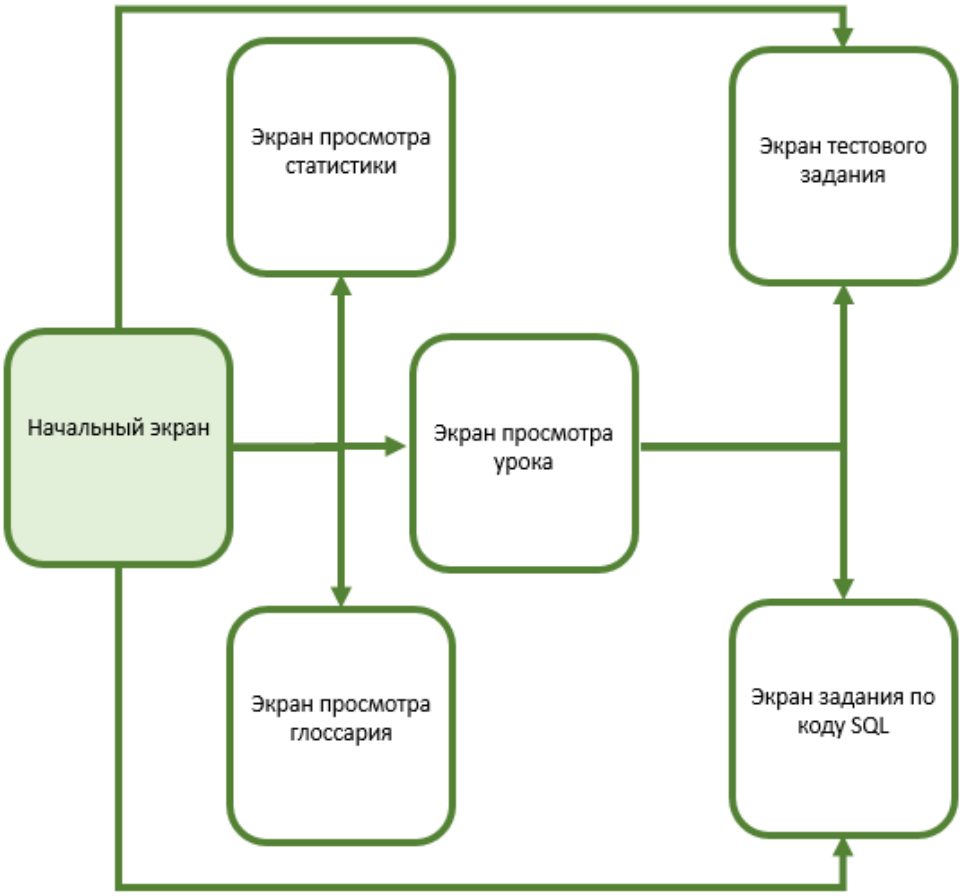

Рисунок 31 – Схема связи экранов приложения

ЗАКЛЮЧЕНИЕ

Целью данной курсовой работы являлось описание структуры, возможностей и функционала мобильного приложения для изучения SQL.

В процессе выполнения курсовой работы была изучена и проанализирована предметная область: основы теории баз данных, их структуры при использовании различных моделей представления данных, актуальность использования и т.д.

Особое внимание уделялось реляционным базам данных и РСУБД, при работе с которыми и используется язык структурированных запросов (SQL). Было произведено описание аналогов SQL-БД, т. н. NoSQL-БД, и сравнение их с реляционными БД.

Также был изучен вопрос об актуальности использования в современном мире мобильных приложений в целом и, в частности, в сфере оказания образовательных услуг. Был произведен поиск и анализ аналогов – готовых мобильных приложений, предоставляющих возможность на бесплатной основе изучить язык запросов SQL.

Были выявлены положительные стороны аналогичных приложений, а также их недостатки, которые были учтены при разработке требований. На основе требований к мобильному приложению была разработана и описана его архитектура, а также произведено проектирование интерфейса.

Так, в результате выполнения курсовой работы было спроектировано мобильное приложение для изучения языка структурированных запросов SQL.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дейт К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 480 с., ил.
2. Грофф Дж. Р., Вайнберг П. Н., Оппель Э. Дж. SQL: полное руководство, 3-е изд. : Пер. с англ. – М. : ООО «И.Д. Вильямс», 2016. – 960 с. : ил. – Парал. тит. англ.
3. Кляйн А., Кляйн Д., Хант В. SQL. Справочник, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2010. – 656 с., ил.
4. Карпова И.П. Базы данных. Курс лекций и материалы для практических заданий. – Учебное пособие. – М.: Питер, 2013. – 240 с.
5. Фуфаев Э.В. Базы данных : учеб. пособие для студ. учреждений сред. проф. образования / Э.В. Фуфаев, Д.Э. Фуфаев. – 7-е изд., стер. – М. : Издательский центр «Академия», 2012. – 320 с.
6. Тарасов С.В. СУБД для программиста. Базы данных изнутри. – М.: СОЛОН-Пресс, 2015. – 320 с., ил.
7. Кара-Ушанов В.Ю. SQL – язык реляционных баз данных : учебное пособие / В.Ю. Кара-Ушанов. – Екатеринбург : Изд-во Урал. ун-та, 2016. – 156 с.
8. Stepik – Лучшие онлайн курсы [Электронный ресурс] – Режим доступа: https://welcome.stepik.org/ru – Дата доступа: 15.11.2022.
9. Sololearn: Learn to Code [Электронный ресурс] – Режим доступа: https://www.sololearn.com/ – Дата доступа: 15.11.2022.
10. Глушенко С.А., Долженко А.И. Разработка мобильных приложений: Учебное пособие – Ростов-на-Дону: издательство РГЭУ (РИНХ), 2018. – 221 с.
11. Черников В.Н. Разработка мобильных приложений на C# для IOS и Android. – М.: ДМК Пресс, 2020. – 188 с.: ил.
12. Льюис Ш., Данн М. Нативная разработка мобильных приложений / пер. с англ. А.Н. Киселева. – М.: ДМК Пресс, 2020. – 376 с.: ил.
13. Статистика мобильных приложений 2021: загрузки, тренды и доходность индустрии [Электронный ресурс] – Режим доступа: https://vc.ru/marketing/245003-statistika-mobilnyh-prilozheniy-2021-zagruzki-trendy-i-dohodnost-industrii – Дата доступа: 27.11.2022.
14. Как работают базы данных в IT: разбор на примерах [Электронный ресурс] – Режим доступа: https://practicum.yandex.ru/blog/chto-takoe-bazy-dannyh/ – Дата доступа: 05.12.2022.
15. SQL и NoSQL. Правда ли одно лучше другого? [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/wunderfund/blog/691178/ – Дата доступа: 08.12.2022.